U. S. Railroad Retirement Board

Application Development Domain Architecture

Application Development Domain Architecture
Table of Contents

Application Development Domain Definition

The Application Development Domain defines the principles, technologies, standards and guidelines for how applications are designed, developed and interact. It enables a high level of system integration, reuse of components, rapid deployment of applications and high responsiveness to changing business requirements.

Our focus is to define principles that enable consistent, high quality development across Application Developments.

High quality systems are efficient, accurate, maintainable, extensible, flexible, scalable and compatible.

Domain Technology Categories

- Languages
- Software Testing and Debugging Tools
- Report Writer Tools
- Multimedia (for CBT)
- Forms Software
- Development Methodologies
- Change Management Software (Repositories)
- Project Management
- Document Management
- Imaging and Document Workflow
- Process Modeling
- PC/Web Development Tools
- COTS Products Integrated Into Developed Applications

Application Development Domain Principles Summary

1. **Products and Technologies:** The RRB will be a “fast follower” in the evaluation, purchase and use of application development products and technologies
2. **Business Requirements:** Business needs/requirements must be established and prioritized before proceeding to application design and development.
3. **Extensible Development:** Applications will be developed that enable adaptability to changes in business needs and technology
4. **Rapid Application Development:** Application development methods and approaches that enable quicker delivery of required, essential functionality will be favored.
5. **N-Tier Design:** Applications will be designed with a minimum of three distinct logical layers consisting of presentation, business rules and data.
6. **Common Components**: Applications will be developed by reusing existing components wherever practical. New components will be developed with reuse in mind.
7. **Validation Process**: Applications will be designed and developed to validate information as close to its source as possible.
8. **Quality Assurance**: Objective feedback, such as IT metrics and survey results, on the quality of an application will be captured periodically and appropriately considered in assessing the success/acceptability of that application.

Domain Relevant Trends

- Timely availability of information (24x7x365) - Increasingly, customers of private industry and government organizations expect timely availability and access to specific, dynamic information and tools.
- No downtime on systems - Internet services and work-at-home programs demand greater availability of systems. Our customers and staff must be able to rely upon our systems to make the services they require available and reliable.
- New presentation and access -The government-wide mandate that services be made available to citizens in as many venues as possible dictates that we learn about and implement up to date presentations of our services and data.
- New security implications - Security requirements are being defined as rapidly as new methods of presenting and collecting data are developed. Changes in the law and other mandates that we must meet have security aspects.
- One and done (client services) - The RRB intends to make it possible for a customer to complete as many governmental transactions as possible in one effort.
- Increased communication - New methods of communication and system design require more frequent and more thorough communication between system owners, users and developers. In addition, the RRB's communications with our constituency must be clear and understandable.
- Faster development/modification - Shorter business cycle times and the success of some organizations in responding to change in the business cycle raise customers' expectations of speed in processing.
- Greater collaboration among RRB organizations and external organizations (Treasury, SSA, Railroads, Unions) - Partnerships are the current model for working relationships with other organizations. This concept implies greater communication, coordination and cooperation, understanding each other's goals and strategies and working toward mutual success.
- Increasing number of development tools – Throughout the IT industry, there is a proliferation in the number of development tools providing new or increased functionality.
- Increasing complexity - Interfacing across Application Developments, providing multiple choices of service venues and reducing paperwork create complex situations. Contention is created trying to achieve all of these goals.
- New and emerging technology standards - XML and Microsoft.Net appear to be emerging development "standards" both in private industry and government organizations, particularly for Web-based development.
- Government Paperwork Elimination Act (GPEA) – The RRB, along with other federal agencies, is required by GPEA to provide services to its customers via the Internet by October 2003.
- Telecommuting - Federal agencies are required by Public Law 106-346 to establish a work at home program for 25% of the federal workforce.

- Web-based applications – The deployment of the Internet has spawned a number of new technologies for developing applications that can operate on this new Application Development. Delivery systems based on IBM 3270 architectures are not easily adapted to this environment.
- Redesign of legacy applications – Because of the requirement that agencies externalize operations for public use, applications designed for internal use must be re-designed to enable direct interactions with the public.
- Use of middleware – This technology arose to facilitate communication between heterogeneous Application Developments that now exist.
- Retirement and Survivors Improvement Act – The need to make applications changes to legacy systems, if the Act is passed, will require resources that would have been assigned to implementing newer technology.
- Section 508 of the Americans with Disabilities Act - Recent legislation and government regulation, such as Section 508, will require RRB systems to incorporate additional considerations into applications design beyond user requirements. These may impact the way RRB gathers requirements and designs applications.
- Push for purchased and contracted software development – In March 2001 OMB directed agencies to compete by October 2002 at least 5 percent of government jobs considered commercial in nature.
- Design and development moving to distributed environment – The IT industry is rapidly developing software techniques that implement application processes across multiple servers and Application Developments, where functions are performed on multiple networked based computing devices.
- Expectation that RRB will exploit newer technologies – The impetus for the RRB to exploit newer technologies, while not driven by the marketplace, is driven by the rising expectations of customers that are aware of other agency's technological advances.
- Moving toward reusable components – As is happening elsewhere in the IT world, RRB mainframe and PC developers have efforts underway to code modules that can be used by multiple applications.

Background of Application Development and Related Technologies at the RRB

The RRB relies heavily on computer applications, developed for the most part by RRB programming staff, to support its core business operations. Since the 1960's, the RRB's catalog of core business applications has grown to over 160 systems, each written to support specific areas of the agency's mission. Aside from a number of administrative PC based applications, most applications were designed to operate on IBM compatible mainframes, located at RRB headquarters, a series of which the RRB has owned over the years. These mainframe applications have no portability to other Application Developments and limited potential connectivity to applications written for the PC or Internet.

The technology that we used for these core processes was considered state-of-the-art until the early 1990's. It was based on procedural programming languages such as COBOL and CA-ADS/O; networked, non-relational databases (CA-IDMS); IBM 3270 data presentation; and batch job operations that process transactions queued from daily business activity. Presently, we continue to automate manual processes by building applications that depend on this model and the related tools that support it.

These legacy applications perform their functions effectively; however, they do not all do so efficiently. They require a significant investment in maintenance, support and operational resources. They evolved over many years, reflecting the capability of the tools of the time as well as the unstructured methods by which we initiate and manage many of our automation initiatives. Enhancements were implemented incrementally, limited by funding and other resource constraints. As a whole, they do not benefit from an efficient “architected” blueprint or design.

The underlying batch operating structure of RRA and RUIA daily, monthly, and annual processing cycles that evolved in the 1960’s remains in effect today. The technological enhancements that were implemented since then improve the way data is captured, processed and presented (use of online 3270 applications rather than forms based data entry) and the way data is stored (mainframe databases rather than flat tape files). These improvements have enhanced employee productivity significantly despite some inherent inefficiency.

The inter-relationships between many of our applications can be characterized as monolithic rather than integrated. Although there have been some improvements in recent years in the way applications communicate, many daily application programs still obtain services from related programs via interconnecting flat files. This design introduces inherent delays.

Recently we have begun to address the technical problems of integrating new PC, LAN, and WEB technologies with these legacy applications for the purpose of servicing core business operations. Our goal clearly is to take advantage of these technologies to enable our core systems to improve service delivery. However, because of the complexities of our legacy systems and their limited adaptability to the new technologies, this will prove to be challenging. It is important to note that we have made important advances in applying the new technology to work processes that are external to but play a supportive role for our core systems. Many PC applications have been developed to help RRB employees accomplish their tasks such as in work tracking and other office functions. The introduction of an imaging system and its associated workflow management software offers significant potential. However, integration of these systems with mainframe operations has been limited because of incompatibilities that must be overcome. Nevertheless, the insights and experience gained from PC development will aid us in solving the technical problems.

The challenges we face are more than merely technical. We will contend with inertia from our past. Our legacy applications were built for use by RRB employees and depend on their expertise to accomplish their functions. Integrating WEB based applications with these systems will raise issues about their suitability for use by our railroad public. The functions that the legacy applications perform are woven into the fabric of our organization’s daily operations. The difficulty of abstracting these functions in order to write smarter systems will influence our ability to adapt the applications to a different environment. For the most part, our automation initiatives and the application systems they produced mirror rather than re-engineer the manual work processes they replaced. Integrating the new technology into our environment will cause us to re-think how we do our work. Finally, the work patterns that we employ in performing our application development tasks will need to change. We face the expectation of more rapid delivery of software product than in the past.

Although our long-term goal may be the replacement of our legacy applications, for the near term, we face the challenge of advancing application development approaches that can leverage the strengths of our legacy systems and integrate them with the new capabilities of new technology. The key word is integration. Whole scale replacement or re-engineering of legacy applications is infeasible: it would be extremely risky and cost prohibitive. We must develop strategies that maintain acceptable performance of existing applications while we exploit the advantages of object oriented languages, networked Application Developments, more rapid development methodologies, and new types of service delivery approaches. Accordingly, the objective of the Applications Development Domain is to develop guiding principles that can transition us from our current state to our target Applications Development Architecture. Whereas the agency has greatly strengthened its information infrastructure with respect to networks, PC desktops, office application suites, server hardware, e-mail, and other productivity tools, little progress has been made thus far in applying the new technology to the software that services our core business processes.

Accordingly, the principles that follow in this document were developed with these issues in mind. They emphasize that a number of key ideas are important in this endeavor:

- Business needs drive the use of technology—not the other way around;
- Adaptability in software design matters—components, reuse and logical layering of functions is paramount if we want to integrate the old with the new;
- Data integrity is a key consideration in application design;
- More rapid production of software products is important when adjusting to change;
- Vigilance in assessing technology developments is required to remain effective;
- Quality is not an after-thought, but part of every stage in the application development process.

Detailed Domain Principles

Domain Principle 1 - Products and Technologies

The RRB will be a "fast follower" in the evaluation, purchase and use of application development products and technologies.

Rationale:

We want to be leading, not "bleeding", edge with emerging technologies to avoid waste inherent in inadvertent "beta testing" of products that are presented as production versions.

We want to avoid forcing agency-wide adoption of products that have not gained acceptance by the IT community at large and, thus, have uncertain life cycles and support.

Timely adoption of new technologies better supports business needs.

The rapid rate of change in information technology requires us to keep pace.

We must meet the expectations of the RRB public and external entities (railroads, other agencies).

Allows RRB to be proactive rather than reactive to new technologies.

May make RRB a more attractive employer and aid in retention of existing IT staff since their skills sets will be more up-to-date.

Implications:

Requires an environment to evaluate the new tools and how they work within the RRB infrastructure that may impact the Platform domain (storage, set-up, interconnections…).

Technologies may be less reliable in earlier releases.

Responsibilities for monitoring products in current use at the RRB must be assigned.

Responsibilities for monitoring and recommending emerging application development technologies must be assigned.

Funding will influence the degree to which the RRB can execute this principle.

Domain Principle 2 - Business Requirements

Establish and prioritize business needs/requirements before proceeding to application design and development.

Rationale

By avoiding design issues in requirements, business needs have greater clarity.

Developers are better able to decide the appropriate tools and technologies considering the enterprise wide technical architecture.

Better chance of meeting user needs, especially the most important requirements if we establish mandatory vs. desirable requirements.

The user will realize benefits more quickly because developers will be able to be more efficient (less rework, waste of developer time).

Makes component development and reuse more likely because we will be better able to identify applicable existing components with the same or similar functionality.

Implications

RRB must revise service request procedures.

Project teams will spend more time in requirements phase.

RRB must train users in identifying requirements and establishing priorities.

RRB must train developers in evaluating requirements and in considering the ability of existing components to meet the requirements.

User and developer must communicate earlier and more frequently throughout development.

High priority requirements must be included in the initial rollout with lower priorities designated for later phase(s).

<u>Domain Principle 3 - Extensible Development</u>

Applications will be developed that enable adaptability to changes in business needs and technology.

Rationale

The maintenance burden will be eased.
Leads to shorter development times for later changes.
Encourages modular design and consequent reuse of components.
Helps avoid any solutions that are not compatible with architectural direction.
May extend viability of associated tools and products.

Implications

Developers must keep up-to-date with technology development.
Requires RRB modernize development skill sets in anticipation of business needs and new technologies, allowing developers to “experiment” with the new knowledge.
Requires better communication of business and technology trends between users and IT staff
The initial development of an application will take longer.
May push RRB to choose an alternative product or technology rather than the “best of breed” to address a particular issue.

<u>Domain Principle 4 – Rapid Application Development</u>

The RRB will favor application development methods and approaches that enable quicker delivery of required, essential functionality.

Rationale

Helps us focus on the most essential requirements first
Provides a better chance of successfully producing a useful product sooner by implementing essential functionality first.
Allows reassessment and modification of requirements without significantly impacting cost and schedule.
Meets user expectations for speed of delivery thereby providing positive reinforcement to users and developers.

Implications

Might have to consider a different System Development Life Cycle (SDLC) model or apply our current SDLC differently.
Requires closer cooperation between users and developers in requirements and design.
Requires more diligence in documentation.
Need to employ “self-documenting” tools.
Requires strong project management practices due to the iterative approach.
Requires training for users and developers.
May produce modular/phased deliverables.

Domain Principle 5 - N-Tier Design

Applications will be designed with a minimum of three distinct logical layers consisting of presentation, business rules and data access.

Rationale

Facilitates reuse by causing functionality/services to be defined more discreetly in components.

Facilitates changes to one layer without impacting others or the addition of new functionality/service, including the addition of another layer/tier.

Promotes greater consistency in applications design since unique logic will most likely be found in the business rules layer.

Enables conversion to web (or other new/unknown) technology.

Promotes greater consistency in applications design since unique logic will most likely be found in the business rules layer.

Produces greater accuracy because functions/services will be centralized in the application and more easily analyzed and tested. Separation of data and business rules will more readily reveal errors and help to isolate them.

Complies with industry best practices.

Implications

Might require us to develop new types of interfaces to legacy systems.

There may be more than 3 layers (e.g. security, middleware, persistence layer, control).

Requires a more structured approach to development and defining requirements.

Boundaries between layers must be carefully defined.

Using tiers will by nature have some impact on performance. We will need to consider this impact when designing and testing applications to ensure that functionality is not impaired.

Developers and definers of business rules must be trained in separating the layers and new testing techniques. The accuracy of the layering must also be tested.

Helps users focus on requirements before design.

May lead to division of labor among developers along the layers/tiers.

Eases application audit and encourages focus on quality.

Eases maintenance.

Domain Principle 6 - Common Components

Applications will be developed by reusing existing components wherever practical. New components will be developed with reuse in mind.

Rationale

Reduces overall complexity of the RRB technology environment by promoting standardized systems interfaces and greater consistency of common functionality/services.

Reduces cost of maintenance and rewrites by minimizing redundancy across various applications that require the same function/services or within an application that uses a function/service multiple times.

Reduces subsequent development time due to reuse of components.

Increases accuracy because all applications will deploy the same tested, proven functionality (components) instead of each application rewriting the functionality/service for their own use.

Allows for faster response to business rule changes that impact multiple applications because changes will only be made in one place.

Common components will be designed to allow change to the component without impacting existing users.

Implications

At a minimum, “component” includes interfaces, modules and objects.
Security implications must be considered
Initial development of common components will be complex and time-consuming
Must develop a commonly available repository to list all common components, their functions and how to access them
Must develop and follow a consistent naming convention
Must define and publish listings and methods of calling common components
Must provide training in methods of building common components
An error in shared modules has greater impact
More coordination among stakeholders needed to develop and implement
Existing use must be considered when changing common components
Must establish authority to change common components and business rules

Principle 7- Validation Process

Applications will be designed and developed to validate information as close to its source as possible.

Rationale

Promotes greater data accuracy by revealing the error when the source data is fresh.
Promotes faster data problem correction.
Limits exception processing after the initial editing.
Allows transaction processing to complete more quickly because the data has been cleansed as early in the process as feasible.
Provides greater likelihood of successfully completing processing.
Provides a high degree of confidence in reporting and analytical processing output.
Data can be shared and integrated (appropriately) across the RRB.

Implications

The steward of data is responsible and accountable for its integrity
Central accessible repository of validation rules must be created
Data owners, users and stewards must agree on what the edit/validation rules are and who is authorized to change them (refer to DATA/OBJECT DOMAIN)
Data owners and stewards must define and document their data (refer to DATA/OBJECT DOMAIN)
Common edit routines must be used to ensure data consistency and integrity
Revalidation is mandatory when data has been transformed or received from an un-trusted source

Principle 8 - Quality Assurance

Objective feedback, such as IT metrics and survey results, on the quality of an application needs to be captured periodically and appropriately considered in assessing the ongoing success/acceptability of that application.

Rationale

- Assures that we are producing quality products both from a usability/business effectiveness perspective as well as through the use of IT efficiency measures.
- Provides objective measurement of how well applications are designed, function and meet the EA principles.
- QA feedback can be used as a mentoring tool for systems development and user/owner/stewards in defining requirements and testing.
- Using the feedback throughout the development life cycle helps build better systems by helping identify previously unidentified requirements or clarifying priorities.
- Ongoing query and analysis of the feedback helps determine the need for upgrade and replacement of systems.
- Improves relations with users by providing a channel for feedback/venting.
- Ongoing quality assurance efforts can aid in measuring efficient use of agency resources.

Implications

- Feedback must be filtered and analyzed using a consistent process and set of criteria.
- Need to establish measurable success factors in cooperation with stakeholders.
- Feedback will help develop best practices and will enable benchmarking.
- Need metrics to measure quality.
- Need QA function, which may require an organizational change.
- Need tools to support the QA function.
- Responsibility for addressing QA findings/recommendations needs to be assigned.
- Multiple sources of feedback will be considered (e.g. operations, users, developers, metrics).

Domain Technology Categories, Standards and Products

This section represents a view of the standards and products that are used to implement solutions.

To better illustrate each technology category, the following attributes of that category are documented when pertinent information exists:

Standards – This table represents standards that have been adopted for central support across the enterprise. Support for a standard may be direct, or indirect by virtue of a supported product that relies heavily on a particular standard. Standards refer to those sanctioned by national or international standards bodies, or industry groups that dictate how products are developed, deployed or interoperate with each other.

Other Standards in Use – This table lists standards that have been adopted by specific areas of the company either directly or by virtue of using a particular product.

Products – This table list those products that:

1. Are centrally supported or used widely across the enterprise
2. Are being considered for central support
3. Have been centrally supported in the past

Other Installed Products – This table lists those products that:

- Are being used by a limited (usually one) number of areas
- Are not centrally supported

Product Lifecycle

Use the following key when referring to the products and standard timelines

Code	Term	Meaning
ID	Identified	The company is aware of the item. It has not been formally evaluated. It is not approved for current use.
RD	Research and Development	Business units may use RD components only as selected participants in a company sanctioned research effort. These items may not be used in any other context and may not be used in production.
P	Pilot	Approved for production, but widespread use not yet encouraged.
I	Invest	Appropriate use encouraged.
M	Maintain	New implementations are not encouraged. Existing systems may continue to rely upon these components and extend existing implementations.
D	Disinvest	In process of being phased out.
O	Obsolete	Vendor or industry support is gone and use in not recommended.
R	Rejected	Investigated and found to not meet the organization’s needs.

THE FOLLOWING TABLES ARE POPULATED WITH PRODUCTS AND STANDARDS, INDICATING LIFECYCLE CATEGORIZATION OVER TIME.

Software Testing and Debugging Tools

Definition: Tools that aid in debugging and testing of software and applications

Standards:

	2001				2002				2003				2004			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
None																

Other Standards in Use:

Standard	Usage Scope
ADP Standards	Mainframe

Products:

	2001				2002				2003				2004			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
Compuware Abend-Aid 9.x	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
Compuware Abend-Aid for IDMS 9.x	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
Compuware Abend-Aid Viewing Facility 9.x	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
IBM COBTEST	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
Viasoft Existing Systems Workbench	MI	M	M	M	M	M	M	M								
ASG Validate 2.x	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
Compuware FileAid 8.x	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
Compuware Data Solutions 3.x	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
ADS Alive	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
Marble DCD III 2.x	M	M	M	M	M	M	M	M								

Other Installed Products:

Product Name	Usage Scope
None	

Languages

Definition: A formal language in which computer programs are written. The definition of a particular language consists of both syntax (how the various symbols of the language may be combined) and semantics (the meaning of the language constructs).

Standards:

	2001				2002				2003				2004			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4

Other Standards in Use:

Standard	Usage Scope

Products:

	2001				2002				2003				2004			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
COBOL	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
Visual Basic	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
Delphi	M	M	M	M	M	M	M	M								
VBScript	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
JavaScript	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
Active Server Pages	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
OPAL Script	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
MS SQL	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
CA-IDMS SQL	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
SAS SQL	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
HTML 4.0	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
XML	ID	ID	ID	R D	R D	R D	R D	R D								
ADS/O	I	I	I	I	I	I	I	I	M	M	M	M	M	M	M	M
CA-IDMS COBOL-Data Manipulation Language (DML)	I	I	I	I	I	I	I	I	M	M	M	M	M	M	M	M
CA-IDMS COBOL-SQL	I	I	I	I	I	I	I	I	M	M	M	M	M	M	M	M
CA-IDMS COBOL-DC	I	I	I	I	I	I	I	I	M	M	M	M	M	M	M	M
COBOL-CICS	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
BMC Control-M	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
dBase	M	M	M	M	M	M	D	D	D	D	D	D	O	O	O	O
MS Access Basic v2	M	M	M	M	D	D	D	D	O	O	O	O	O	O	O	O
Lotus 123	D	D	D	D	D	D	D	D	O	O	O	O	O	O	O	O
Paradox v9	M	M	M	M	M	M	M	M	D	D	D	D	D	D	D	D
Paradox v7	D	D	D	D	D	D	D	D	O	O	O	O	O	O	O	O
Visual Basic for Applications (VBA)	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
IBM High-Level Assembler v2.x	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
CA-EZTrieve+ v6	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
REXX	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
CList	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
SAS	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
JCL	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
Microfocus COBOL	M	M	M	M	M	M	M	M	D	D	D	D	O	O	O	O
Visual Basic.net	ID	ID	ID	R	R	R	I	I	I	I	I	I	I	I	I	I

	2001				2002				2003				2004			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
				D	D	D										
TSI International Keymaster v6.x	M	M	M	M	D	D	D	D	D	D	D	D	O	O	O	O
Visual C++	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
Visual J++	ID	ID	ID	ID	ID	ID	ID	ID								
Visual Fox Pro	ID	ID	ID	ID	ID	ID	ID	ID								
Syncsort 3.x	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
Group 1 Code 1+ with Zip+4 2.x	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
Group 1 Barcode 2.x	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I

Other Installed Products:

Product Name	**Usage Scope**
Rbase	BFO (5 users)
Decision Technology Analyzer 4.x	BFO

Multimedia

Definition: Computer systems that integrate audio, video and data.

Standards:

	2001				2002				2003				2004			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
None																

Other Standards in Use:

Standard	Usage Scope
None	

Products:

	2001				2002				2003				2004			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
Authorware	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
Fireworks	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
Adobe Photoshop	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I

Other Installed Products:

Product Name	Usage Scope

Forms Software

Definition: Products that aid in the development of forms and reports.

Standards:

	2001				2002				2003				2004			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
None																

Other Standards in Use:

Standard	**Usage Scope**
None	

Products:

	2001				2002				2003				2004			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
Xerox HFDL 3.x	I	I	I	I	D	D	D	D								
OMNIFORM Developer	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
Paris (XL Print)			RD	RD	RD											
Elixir			RD	RD	RD											

Other Installed Products:

Product Name	**Usage Scope**
JetForm Formflow	BIS/IMD

Development Methodologies

Definition: Technologies related to software development methodology such as methods, notations and tools.

Standards:

	2001				2002				2003				2004			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
Unified Modeling Language	RD	RD	RD	RD	I	I	I	I	I	I	I	I	I	I	I	I
Unified Process	RD	RD	P	P	I	I	I	I	I	I	I	I	I	I	I	I

Other Standards in Use:

Standard	**Usage Scope**
RRB's SDLC	Universal as applicable

Products:

	2001				2002				2003				2004			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
Popkin Systems Architect	RD	RD	RD	RD												
MS Visual Modeler	RD	RD	RD	RD	I	I	I	I	I	I	I	I	I	I	I	I
MS Visio	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I

Other Installed Products:

Product Name	**Usage Scope**
Optima, Inc. Structures 3.x	Obsolete (Warnier-Orr Charts)

Project Management Software

Definition: Products that track and manage projects throughout the SDLC

Standards:

	2001				2002				2003				2004			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
None																

Other Standards in Use:

Standard	Usage Scope
None	

Products:

	2001				2002				2003				2004			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
AGS Wings 1.x	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
MS Project 97	M	M	M	M	M	M	D	D	D	D						
MS Project 2000	RD	RD	RD	RD	RD	RD										

Other Installed Products:

Product Name	Usage Scope

Report Writers

Definition: Products that produce reports with minimal programming

Standards:

	2001				2002				2003				2004			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
None																

Other Standards in Use:

Standard	Usage Scope
None	

Products:

	2001				2002				2003				2004			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
SAS Enterprise Reporter (PC)	P	P	P	P												
Crystal Reports (PC)	R D	R D	D	D												

Other Installed Products:

Product Name	Usage Scope
CA-JARS 7.x	1 user (AOS) - **DISINVEST**

Change Management Software

Definition: Products that are used to identify, organize and manage changes to software built by programmers

Standards:

	2001				2002				2003				2004			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
None																

Other Standards in Use:

Standard	Usage Scope
None	

Products:

	2001				2002				2003				2004			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
CA-Endevor	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
CA-IDMS IDD	I	I	I	I	I	I	I	I	M	M	M	M	M	M	M	M
MS SourceSafe	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
Panvalet	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M

Other Installed Products:

Product Name	Usage Scope

Document Publishing

Definition: Tools for electronically storing and publishing documents

Standards:

	2001				2002				2003				2004			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
None																

Other Standards in Use:

Standard	Usage Scope
None	

Products:

	2001				2002				2003				2004			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
HSL	D	D	D	D	D	D	D	D								
Document Sciences TextBook 3.x	M	M	M	M	D	D	D	D								
IBM BookManager	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
Adobe Acrobat	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
MS Outlook Exchange	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
Document Director	M	M	M	M	D	D	D	D								
MS Publisher	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I

Other Installed Products:

Product Name	Usage Scope

Imaging and Document Workflow

Definition: Products that enable digital images and computer output to be stored, organized and routed into the workflow

Standards:

	2001				2002				2003				2004			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
None																

Other Standards in Use:

Standard	Usage Scope
None	

Products:

	2001				2002				2003				2004			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
Eastman Document MGT Workstation	I	I	I	I	I	I	I	I	D	D	D	D	D	D	D	D
Eastman Form Builder	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
Eastman Workdesk	RD	RD	RD	RD	RD	RD	RD	RD	I	I	I	I	I	I	I	I
Eastman Web Connector	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
Eastman Imaging for Windows	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
Eastman Route Builder	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I

Other Installed Products:

Product Name	Usage Scope
WinSpace	100 Licenses-for people requiring a larger desktop for imaging work
Imaging Professional	100 Licenses-for people with early release Windows 95 operating system

PC/Web Development

Definition: Software tools for the development of PC and Web based applications

Standards:

	2001				2002				2003				2004			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
HTML 4.0	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
XML	ID	ID	ID	RD	RD	RD	RD	RD								
ADA Sect. 508	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I

Other Standards in Use:

Standard	**Usage Scope**
MS Front Page	Universal
MS Visual Studio	Universal
MS Internet Information Server	Universal

Products:

	2001				2002				2003				2004			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
MS Front Page	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
MS Visual Studio	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
MS Visual Studio.Net				RD	RD	RD	RD	I	I	I	I	I	I	I	I	I
MS Transaction Server	ID	ID	I	I	I	I	I	I	I	I	I	I	I	I	I	I
MS Message Queue Server	ID	ID	ID	ID	RD	RD	RD	RD								
MS Internet Explorer 4.x	M	M	D	D	D	D	D	D								
MS Internet Explorer 5.x	M	M	M	M	M	M	M	M	D	D	D	D	D	D	D	D
MS Internet Explorer 5.5	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
SAS(PC)	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I

Other Installed Products:

Product Name	**Usage Scope**
MacroMedia Dreamweaver	1 user (CIO)
Lynx Browser	Webmaster (Section 508 compatibility tester)
Netscape	Webmaster (Testing multi-browser compatibility)
CA Jasmine	One copy in Programs

Commercial Off the Shelf (COTS) Products

Definition: COTS products integrated into developed applications

Standards:

	2001				2002				2003				2004			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
None																

Other Standards in Use:

Standard	**Usage Scope**
None	

Products:

	2001				2002				2003				2004			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
MS Office 2000	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
MS Office 97	I	I	I	I	M	M	M	M	M	M	D	D	D	D	D	D
Group 1 Code 1	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
MS Outlook	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I

Other Installed Products:

Product Name	**Usage Scope**

Pattern 1

Component Development

Purpose

To provide direction for creating discrete units of reusable software that provides a specific service or logically grouped services

Applicability

Create discrete units of reusable software when:

- A service could be used by other applications;
- A service will be used multiple times within an application;
- A standard interface into a business service is desired. There can be many services that are combined to provide a business service within a component. Each of these services could be accessed directly;
- A service is subject to frequent change;
- A tiered architecture is desirable
- And, for changes to existing systems to build new components, stakeholders agree on any impact on the timely completion of the project

Assumptions

Use of Component Design assumes:

- Interrelated rules can be effectively captured and combined to define a service within a component
- Business rules owners agree on the rules
- In developing components for reuse opportunity must be granted for iterative development until skills in defining common requirements are achieved.

Structure Overview

None at this time

Detailed Pattern Description

The details of this pattern will be developed as part of the "to be" architecture.

Benefits

Reduces overall complexity of the RRB technology environment by promoting greater consistency of common functionality/services and standardized systems interfaces
Reduces cost of maintenance and rewrites because changes will only be made to the common component instead of having to change each application
Minimizes redundancy
Increases accuracy because we avoid the risk of having multiple modules perform the same function differently
Allows for faster response to business rule changes
Common components will be designed to allow change to the component without impacting existing users
After initial development, meets user expectations for speed of delivery by reducing subsequent development time due to reuse of components, thereby providing positive reinforcement to users and developers

Allows developers to focus their creativity on unique processes rather than repetitive business functions

Consequences

When revising an existing system, converting functions to modules may take longer
Initial development of components may take longer
Error in the component has greater impact
Duplication of effort can occur if components are created but not reused
Development staff will need training on component
There may be system performance impact

Related Patterns

Tiered Development

Known Uses

Internet services project (Scott Palmer)

PREH edit modules (Melissa Mickelberry)

Pattern 2

System Development Life Cycle (SDLC)

Purpose

To direct the choice of an SDLC model that is appropriate to the type of project

Applicability

Apply the SDLC pattern to all development regardless of Application Development or extent of development effort. The pattern applies to agency and contractor staff.

Assumptions

Sufficient information is available at project definition to determine the most advantageous model to use.

Structure Overview

UNDER DEVELOPMENT

Detailed Pattern Description

Consider:

- Iterative, spiral and linear examples
- Emergency fixes
- Small jobs
- Equivalent life cycles proposed by contractors
- Testing
- QA

Benefits

1. Provides standardization for all development work
2. Formalizes current practice for small projects
3. Shortens the life cycle where appropriate
4. Establishes the division of duties
5. Provides controls that are appropriate to the scope of the project
6. Makes development more efficient and effective

Consequences

1. Misuse of models
2. If requirements change mid-stream, the model may not be the best fit.
3. Training will be required in the use of the models and the pattern.
4. Conflict may arise among stakeholders over the choice of the model.
5. Initial implementation may cause confusion and slow progress

Variations

Under development

Related Patterns

Under development

Known Uses

Under development

Pattern 3

Tiered Development

Purpose

Divide systems into tiers that reduce complexity and facilitate reuse, integration, flexibility and division of labor

Applicability

- Use tiered development for all new developments
- Consider tiered development for maintenance changes depending on extent
- Consider converting legacy systems to tiered development if the agency plans on investing in the application

Assumptions

It is assumed that:

- Some level of partitioning of the application is possible
- Tools and infrastructure allow tiered development

Structure Overview

UNDER DEVELOPMENT

Detailed Pattern Description

Under development

Benefits

1. Separation of duties; simultaneous development of tiers may be possible
2. Facilitate Reuse
3. Easier and quicker changes to one layer without impacting others
4. Greater consistency in applications design
5. Enables conversion to web (or other new/unknown) technology
6. Promotes better management of data and information as enterprise-wide assets
7. Greater accuracy
8. Complies with industry best practices
9. Facilitates flexible system partitioning; individual layers can be implemented on more than one server

Consequences

1. Could reduce system performance
2. Major paradigm shift in RRB requirements definition
3. Training in tiered design will be required for users and developers

Variations

Under development

Related Patterns

Under development

Known Uses

Under development

Domain Participants

Domain Team Leader: Leon Roose

Line of Business Representatives: Linda Osman, Jim Verplaetse

Domain Participants: Jim Balla, Barbara Gleason, Les Gunther, Patricia Henaghan ,Bob Kinsella, Judy Lombardo, Shannon O'Hara, Scott Palmer

OEA Representative: Deborah Carter

Appendix 1: Domain Glossary

Term	Definition
N-Tier	In the multi-tier (or N-tier) model, separate software layers are defined to perform discrete logical functions. For example, in a PC/Server environment an application server is used to manage the business logic (e.g. a COM object programmed to compute a specific PIA), a database server handles access logic, and the desktop computer and browsers handle the presentation logic. This structure provides for a business tier, a data tier, and a presentation tier. The logical separation of the tiers allows for simple, straightforward modifications to each of the tiers without undue impact on the others.
Data/Information	Data can be considered as raw facts, numbers or identifiers which become information when interpreted in the context of the use of the data.
"As close to the source as possible"	Editing data "as close to the source as possible" means editing data whenever it is created or modified. The same piece of data may need to be edited more than once. For example, "monthly check rate" may be created in ROC or SURPASS, and should be edited there. It will be passed on to the Daisy system, where it may be changed to subtract Medicare Because it has been changed, it should be edited in the Daisy system. It then goes to the Tax system, where it may be changed again to subtract taxes. The source of the change, the Tax system, should edit the field. Subsequent systems, such as PREH, which do not alter the data, should not have to edit if the previous systems have all done their editing appropriately.

Appendix 2: Conceptual to Domain Principle Matrix

	Relationship Between RRB's Domain Principles And Conceptual Architecture Principles																								
Domain Principle	***Conceptual Architecture Principles***																								
	CA1	CA2	CA3	CA4	CA5	CA6	CA7	CA8	CA9	CA10	CA11	CA12	CA13	CA14	CA15	CA16	CA17	CA18	CA19	CA20	CA21	CA22	CA23	CA24	CA25
D-1				X					X	X			X					X					X		
D-2				X																	🟩	X	X		
D-3				X	X	X						X						X	X						
D-4					X				X			X													
D-5					X	X	X	🟥	X					X				X							
D-6					X	X	X					X		X				X			X				
D-7							X	X																	
D-8																		X				X			X

<u>Conceptual Architecture Guiding Principles:</u>
1. Use guidelines consistent with the Federal Enterprise Architecture. 2. Support a single Enterprise Wide Technical Architecture (EWTA). 3. IT projects are to be consistent with the Enterprise Architecture. 4. Business processes drive technical architecture. 5. Reduce integration complexity. 6. Technical architecture must be extensible and scalable. 7. Manage information and data as enterprise-wide assets. 8. Validate information as close to its source as possible. 9. Enhance the ability to capitalize on and exploit business information. 10. Support multiple data types. 11. Make an informed buy versus lease versus build decision before proceeding with any new development project. 12. Require shorter development cycle times. 13. Keep current with emerging technologies and their applicability to enterprise architecture. 14. Maximize infrastructure asset reuse. 15. Sustain reliable connectivity. 16. IT systems will be implemented in adherence with the agency's security, confidentiality and privacy policies. 17. The agency will use a consistent set of security interfaces and procedures. 18. Reduce total cost of operation (TCO). 19. Extend E-Mail to Become a Corporate Information Exchange Vehicle. 20. Adopt Open Systems Standards. 21. Reduce duplicate information systems. 22. Consider impact on business partners. 23. Maximize and exploit Internet and Intranet technologies and approaches. 24. Integrate Enterprise Architecture into the investment management process. 25. Customer perception is a measure of the quality of the automation processes.